Journal of Korean Institute of Intelligent Systems
Vol. 25, No. 5, October 2015, pp. 515-521
http://dx.doi.org/10.5391/JKIIS.2015.25.5.515

ISSN(Print) 1976-9172
ISSN(Online) 2288-2324

JKIIS

SymMerge 알고리즘의 새로운 복잡도 분석

A New Complexity Analysis of the SymMerge Algorithm

김복선
Pok-Son Kim

국민대학교 수학과
Department of Mathematics, Kookmin University

요 약

SymMerge 알고리즘은 두 입력수열 u와 v ($|u|=m$, $|v|=n, m \le n$)에 대한 효율적 병합 알고리즘이다. SymMerge 알고리즘의 비교횟수와 관련한 복잡도 분석을 하고자 하며 지금까지의 복잡도 분석은 복잡도의 상계값을 찾으므로 점근적 계산방법을 통해 이루어졌다. 이 논문에서는 지금까지의 분석방법과는 달리 SymMerge 알고리즘의 대표적 두 special case에 해당하는 "Symmetric case"와 "Maximum spanning case"에 있어서 병합을 위해 요구되는 정확한 비교횟수를 즉 비교횟수의 최소상계 값을 계산해 보이고자 한다. "Symmetric case"의 경우 사이즈 $m=2^k, n=2^l, l \ge k$인 임의의 입력수열에 대해 SymMerge 알고리즘이 필요로 하는 비교횟수는 정확하게 $m\log\frac{n}{m}+4m-\log m-3$이고 "Maximum spanning case"의 경우 사이즈 $m=2^k, n=2^m-m$인 임의의 입력수열에 대해 SymMerge 알고리즘이 필요로 하는 비교횟수는 정확하게 $\frac{1}{2}m^2+(m+1)\log m-\frac{3}{2}m+2$임을 계산해 보인다. 추가로 이들 두 special case에 있어서 요구되는 비교횟수가 재귀적 함수에 의해 정의될 수 있음을 보인다.

키워드 : 머징 알고리즘, worst case complexity (키워드 수는 5개 정도가 적당)

Abstract

The SymMerge algorithm is an efficient merging algorithm for input sequences u and v of sizes $|u|=m$ and $|v|=n, m \le n$. We consider complexity analysis for SymMerge algorithm regarding to the required number of comparisons. The focus of the previous complexity analysis was on finding the values of upper bounds, i.e. showing the asymptotical optimality. In this paper, in a different way from the previous complexity analysis, we show that the overall required number of comparisons for two representative special cases "symmetric case" and "maximum spanning case" can be calculated exactly i.e. the least upper bounds regarding to the required number of comparisons are calculated. Symmerge requires exactly $m\log\frac{n}{m}+4m-\log m-3$ comparisons for symmetric case of sizes $m=2^k, n=2^l, l \ge k$ of input sequences and exactly $\frac{1}{2}m^2+(m+1)\log m-\frac{3}{2}m+2$ comparisons for maximum spanning case of sizes $m=2^k, n=2^m-m$ of input sequences. Additionally we show that the complexity of the Symmerge algorithm regarding to the overall required number of comparisons for these special cases can be defined by recurrence relations.

Key Words : Merging algorithm, Complexity analysis, Recurrence relation.

Received: Mar. 22, 2015
Revised : Apr. 5, 2015
Accepted: Sep. 23, 2015
†Corresponding author
pskim@kookmin.ac.kr

1. Introduction

Merging denotes the operation of rearranging the elements of two adjacent sorted sequences of sizes m and n, so that the result forms one sorted sequence of $m+n$ elements.

Recent work in this area are the publications [2-4], that describe algorithms which are all asymptotically optimal regarding the number of comparisons as well as assignments. However, these algorithms are structurally quite complex.

In [5] we presented a stable minimum storage merging algorithm called SymMerge and investigated its worst case complexity regarding the number of comparisons as well as assignments. The focus of the complexity analysis was on finding the values of upper bounds, i.e. showing the asymptotical optimality. We considered the set consisting of all merging pairs (nodes) which arise during the computation. Then we partitioned the set into $k+1$ recursion groups such that each recursion group i, $i=0,1,2,\cdots k$ has at most 2^i pairs. For each recursion group the value of upper bound for the required number of comparisons was calculated. So, for all $k+1$ recursion groups the value of upper bound for the required number of comparisons could be calculated and showed the asymptotical optimality of the SymMerge algorithm.

In this paper we consider complexity analysis for two special cases of input sequences of SymMerge "symmetric case" and "maximum spanning case" in a different way from the previous complexity analysis. We show that the overall required number of comparisons for these special cases can be calculated exactly i.e. the least upper bounds regarding to the required number of comparisons are calculated. Symmerge requires exactly $m\log\frac{n}{m}+4m-\log m-3$ comparisons for symmetric case of sizes $m=2^k, n=2^l, l\geq k$ of input sequences and exactly $\frac{1}{2}m^2+(m+1)\log m-\frac{3}{2}m+2$ comparisons for maximum spanning case of sizes $m=2^k, n=2^m-m$ of input sequences.

Additionally we show that the complexity of the Symmerge algorithm regarding to the overall required number of comparisons for these special cases can be defined by recurrence relations.

2. SymMerge Algorithm

We start with a brief introduction of our approach to merging [5, 6]. Let u and v be two adjacent ascending sorted sequences with $|u|\leq|v|$. We decompose the longer sequence v into three parts v_1wv_2 such that $|w|=|u|$ and either $|v_2|=|v_1|$ or $|v_2|=|v_1|+1$. Then we start at the leftmost element in u and at the rightmost element in w and compare the elements at these positions. We continue doing so by symmetrically comparing element-pairs from the outsides to the inside. There can occur at most one position, where the relation between the compared elements alters from 'not greater' to 'greater'. So we have found the bounds for a rotation, i.e. side-changing elements. Due to this technique of symmetric comparisons we call our algorithm SymMerge. Moreover the computation for finding the bounds for a rotation may also happen in the style of a binary search. Then only $\lfloor\log(\min(|u|,|v|))\rfloor+1$ comparisons are necessary. By recursive application of this technique to the arising subsequences we get a sorted result.

Now we describe the SymMerge algorithm formally. We dene $u\leq v$ $(u<v)$ iff $x\leq y$ $(x<y)$ for all elements $x\in u$ and for all elements $y\in v$.

We merge u and v as follows:

If $|u|\leq|v|$, then

(a1) we decompose v into v_1wv_2 such that $|w|=|u|$ and either $|v_2|=|v_1|$ or $|v_2|=|v_1|+1$.

(a2) we decompose u into u_1u_2 $(|u_1|\geq 0,\ |u_2|\geq 0)$ and w into w_1w_2 $(|w_1|\geq 0,\ |w_2|\geq 0)$ such that $|u_1|=|w_2|$, $|u_2|=|w_1|$ and $|u_1|\leq|w_2|$, $u_2>w_1$.

(a3) we rotate $u_2v_1w_1$ to $v_1w_1u_2$.

(a4) we recursively merge u_1 with v_1w_1 as well as u_2 with w_2v_2. Let u' and v' be the resulting sequences, respectively.

else

(b1) we decompose u into u_1wu_2 such that $|w|=|v|$ and either $|u_2|=|u_1|$ or $|u_2|=|u_1|+1$.

(b2) we decompose v into v_1v_2 $(|v_1|\geq 0,\ |v_2|\geq 0)$ and w into w_1w_2 $(|w_1|\geq 0,\ |w_2|\geq 0)$ such that $|v_1|=|w_2|$, $|v_2|=|w_1|$ and $|w_1|\leq|v_2|$, $w_2>v_1$.

(b3) we rotate $w_2u_2v_1$ to $v_1w_2u_2$.

(b4) we recursively merge u_1w_1 with v_1 as well as w_2u_2 with v_2. Let u' and v' be the resulting sequences, respectively.

$u'v'$ then contains all elements of u and v in sorted order.

3. Complexity Analysis

In this section we consider complexity analysis regarding the number of comparisons. Unless stated otherwise, let us denote $m=|u|$, $n=|v|$, $m\leq n$, $k=\lfloor\log m\rfloor$. Further let m_j^i and n_j^i denote the sizes of sequences merged on the ith recursion level where the index j denotes the order of the merged sequences. Initially (on the recursion level 0), it holds $m_1^0=m$ and $n_1^0=n$. On the next recursion level 1, (m_1^0,n_1^0) is divided into two pairs (children nodes). We denote these by (m_1^1,n_1^1), (m_2^1,n_2^1)

where the sequences of lengths m_1^1 and m_2^1 are merged with the sequences of lengths n_1^1 and n_2^1 respectively. On the recursion level i, the sequences of lengths $m_1^i, m_2^i, m_3^i, \cdots$ and m_j^i are merged with the sequences of lengths $n_1^i, n_2^i, n_3^i, \cdots$ and n_j^i respectively (with $2 \le j \le 2^i$). In the following let (m_j^i, n_j^i) denote *a merging pair* in which the size of the left (right) sequence is $m_j^i\,(n_j^i)$. As result of the decomposition process we get such merging pairs. Accordingly let (m,n) denote the pair given as an input.

For each input pair (m,n), $m \le n$, the SymMerge algorithm generates a binary tree in which each node corresponds to a merging pair (m_j^i, n_j^i). We call such a binary tree *a decomposition tree*.

Till now we have considered complexity analysis of the SymMerge algorithm using recursion group [6]. We have partitioned decomposition trees into several recursion groups. In this paper we consider 2 different special cases generated by the SymMerge algorithm and have look at a new complexity analysis for these special cases regarding the number of comparisons.

Symmetric Case

There are input pairs (m,n) such that in their decomposition trees every merging pair (subsequence merging) always triggers two nonempty merging pairs. So, if $m = 2^k$, then there are exactly 2^i merging pairs on each recursion level $i, 0 \le i \le \log m$. We call this special case *symmetric case*. Fig. 1 shows an example of symmetric case for input pair $(8,8)$.

Lemma 1. ([1] Lemma 3.1) If $k = \sum_{j=1}^{2^i} k_j$ for any $k_j > 0$ and integer $i \ge 0$, then $\sum_{j=1}^{2^i} \log k_j \le 2^i \log \frac{k}{2^i}$.

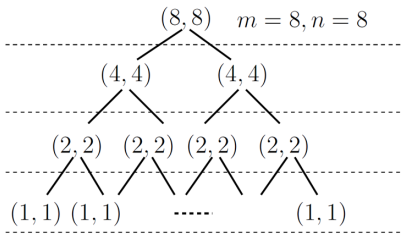

그림 1. Symmetric case의 예

Fig. 1. An example of symmetric case

Theorem 2. Given input pair of size (m,n) of symmetric case and let $m = 2^k, n = 2^l, l \ge k$, then the required number of comparisons on the recursion level k is $2^k(\lfloor \log \frac{n}{2^k} \rfloor + 1) = 2^k(\lfloor \log 2^{l-k} \rfloor + 1) = (l-k+1)2^k$.

Proof. In symmetric case every merging pair (subsequence merging) always triggers two nonempty merging pairs and the size of each pair is divided in half. For example, the size n of the right part of the merging pair is halved on the recursion level 0. Therefore, on the recursion level k there are 2^k merging pairs altogether and each merging pair has the size $(1, n/2^k) = (1, 2^l/2^k) = (1, 2^{l-k})$. Hence the required number of comparisons on the recursion level k is $2^k(\lfloor \log 2^{l-k} \rfloor + 1) = (l-k+1)2^k$ □

Theorem 3. If $m = 2^k, n = 2^l, l \ge k$, then the complexity of the SymMerge algorithm has the least upper bound $m \log \frac{n}{m} + 4m - \log m - 3$ for any input pair (m,n) of the symmetric case.

Proof. The binary search of recursion level 0 requires $\log m + 1$ comparisons. For each recursion level $i, (i = 1, \cdots, k-1)$ we need $2^i(\log \frac{m}{2^i} + 1)$ comparisons. On the last recursion level k we need $\sum_{i=1}^{2^k} (\log n_i^k + 1)$

$= \sum_{i=1}^{2^k} (\log \frac{n}{m} + 1) = \sum_{i=1}^{2^k} (\log 2^{l-k} + 1) = 2^k(l-k) + 2^k$

comparisons. So the overall number of comparisons for all $k+1$ recursion levels is $\log m + 1 + \sum_{i=1}^{k-1} 2^i(\log \frac{m}{2^i} + 1)$

$+ \sum_{i=1}^{2^k} (\log n_i^k + 1) = \log m + 1 + (2^k - 2)\log m - \sum_{i=1}^{k-1} i2^i$

$+ 2^k - 2 + 2^k(l-k) + 2^k$. Since $\sum_{i=1}^{k-1} i2^i = (k-2)2^k + 2$, the complexity of the SymMerge algorithm has the least upper bound (최소상계) $\log m + 1 + (2^k - 2)\log m$

$-((k-2)2^k + 2) + 2^k - 2 + 2^k \log \frac{n}{m} + 2^k = m \log \frac{n}{m} + 4m$

$- \log m - 3$. □

Maximum Spanning Case

There are input pairs (m,n) that have the maximal recursion depth m. In this case m is decomposed into either $(1(=m_1^1), n_1^1)$ and $(m-1(=m_2^1), n_2^1)$ or $(m-1(=m_1^1), n_1^1)$ and $(1(=m_2^1), n_2^1)$. Without loss of generality we suppose that (m,n) is decomposed into $(1(=m_1^1), n_1^1)$ and $(m-1(=m_2^1), n_2^1)$ and each $(m_2^i, n_2^i), i = 1,2,\cdots,m-2$ is decomposed into $(1(=m_1^{i+1}), n_1^{i+1})$ and $(m_2^i - 1(=m_2^{i+1}), n_2^{i+1})$. We call this case *maximum spanning case*. Fig. 2 shows a decomposition tree for max-

imum spanning case.

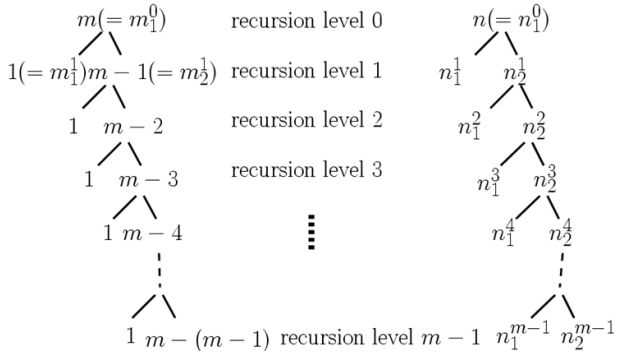

그림 2. Maximum Spanning의 경우

Figure 2. Maximum spanning case

We have the following lemma.

Lemma 4. $\sum_{i=2}^{2^k}(\lfloor \log i \rfloor +1) = k+2^k(k-1)+1$.

Proof. It holds $\sum_{i=2}^{2^k}(\lfloor \log i \rfloor +1) = k+\sum_{i=1}^{k-1} i2^i + 2^k - 1$.

Since $\sum_{i=1}^{k-1} i2^i = (k-2)2^k+2$, it holds $\sum_{i=2}^{2^k}(\lfloor \log i \rfloor +1)$

$= k+(k-2)2^k+2+2^k-1 = k+2^k(k-1)+1$. □

Theorem 5. If $m=2^k$, then the complexity of the SymMerge algorithm has the upper bound $m\log\frac{n}{m}$ $+(m+1)\log m+1$ for any input sequence (m,n) of the maximum spanning case.

Proof. The binary search of recursion level 0 requires $\log m+1$ comparisons. On each recursion level i, $(1 \le i \le m-2)$ we need $\lfloor \log n_1^i \rfloor +1+ \lfloor \log(m-i) \rfloor$ $+1$ comparisons. For the final recursion level $m-1$ we need $\lfloor \log n_1^{m-1} \rfloor +1+ \lfloor \log n_2^{m-1} \rfloor +1$ comparisons. Here, it holds $\sum_{i=1}^{m-1} n_1^i + n_2^{m-1} = n$. So the overall number of comparisons for all m recursion levels is equal to $\sum_{i=1}^{m-1}(\lfloor \log n_1^i \rfloor +1) + \lfloor \log n_2^{m-1} \rfloor +1 + \sum_{i=2}^{m}(\lfloor \log i \rfloor +1)$.

Since $\sum_{i=1}^{m-1} n_1^i + n_2^{m-1} = n$ and by lemma 1 and lemma 2 , SymMerge needs at most $m\log\frac{n}{m}+m+\log m$ $+m(\log m-1)+1 = m\log\frac{n}{m}+(m+1)\log m+1$ comparisons for the maximum spanning case. □

Now we compare the least upper bound $m\log\frac{n}{m}+4m-\log m-3$ for the symmetric case with the upper bound $m\log\frac{n}{m}+(m+1)\log m+1$ for the maximum spanning case. We have $m\log\frac{n}{m}$ $+(m+1)\log m+1-(m\log\frac{n}{m}+4m-\log m-3)$ $= m\log m+2\log m-4m+4 \ge 0$ for all $m=2^k$. This means that the complexity of the SymMerge algorithm has the upper bound $m\log\frac{n}{m}+(m+1)\log m+1$ for any input pair (m,n) of the symmetric case. So the following corollary holds.

Corollary 6. If $m=2^k$, then the complexity of the SymMerge algorithm has the upper bound $m\log\frac{n}{m}$ $+(m+1)\log m+1$ for the symmetric case and maximum spanning case.

Additionally corollary 6 can be explained as follows:
The overall required number of comparisons from recursion level 0 to recursion level $k-1$ for the symmetric case is $\log m+1+\sum_{i=1}^{k-1} 2^i(\log\frac{m}{2^i}+1)$. Here, the number of the considered merging pairs is $1+2+\cdots+2^{k-1} = 2^k-1$. The overall required number of comparisons for the merging pairs $(m_1^0,n_1^0),(m_2^1,n_2^1),(m_2^2,n_2^2),\cdots,(m_2^{m-2},n_2^{m-2})$ of the maximum spanning case is $\sum_{i=2}^{m}(\lfloor \log i \rfloor +1)$. Here, the number of the considered merging pairs is $m-1=2^k-1$ as well. However we know it holds $\log m+1+$ $\sum_{i=1}^{k-1} 2^i(\log\frac{m}{2^i}+1) \le \sum_{i=2}^{m}(\lfloor \log i \rfloor +1)$.

Further, the required number of comparisons for the recursion level k of the symmetric case is $\sum_{i=1}^{2^k}(\log n_i^k+1)$ $= 2^k\left\lfloor \log\frac{n}{m} \right\rfloor + 2^k$. The required number of comparisons for the remaining part of the maximum spanning case is $\sum_{i=1}^{m-1}(\lfloor \log n_1^i \rfloor +1) + \lfloor \log n_2^{m-1} \rfloor +1$ where it holds $\sum_{i=1}^{m-1} n_1^i + n_2^{m-1} = n$. By lemma 1, we have $\sum_{i=1}^{m-1}(\lfloor \log n_1^i \rfloor +1) + \lfloor \log n_2^{m-1} \rfloor +1 \le 2^k\left\lfloor \log\frac{n}{m} \right\rfloor + 2^k$.

Hence, the least upper bound $m\log\frac{n}{m}+4m-\log m-3$

for the symmetric case is equal or less than $m\log\frac{n}{m}+(m+1)\log m+1$ which corresponds to upper bound of the maximum spanning case.

In [7] we considered the following relationship between m and n of input pair (m,n) for maximum spanning case.

Theorem 7. [7] Let any input pair (m,n) of maximum spanning case be given. Then (m,n) satisfies the relationship between m and n such that $n \geq 2^m - m$.

Theorem 8. Let any input pair (m,n), $n=2^m-m$ of maximum spanning case be given. Then $(m-i,2^{m-i}-(m-i))$ is decomposed into $(1,2^{m-(i+1)}-1)$ and $(m-(i+1),2^{m-(i+1)}-(m-(i+1)))$ on each recursion level $i, i=0,1,\cdots,m-2$.

Proof. Since input pair $(m,2^m-m)$ corresponds to maximum spanning case, $(m,2^m-m)$ is decomposed into $(1(=m_1^1),n_1^1)$ and $(m-1(=m_2^1),n_2^1)$ on the recursion level 0 where it holds $n_1^1=\frac{2^m-m-m}{2}+(m-1)=2^{m-1}-1$ and $n_2^1=\frac{2^m-m-m}{2}+1=2^{m-1}-(m-1)$. By the identical calculation method $(m-1,2^{m-1}-(m-1))$ is decomposed into $(1,2^{m-2}-1)$ and $(m-2,2^{m-2}-(m-2))$ on the recursion level 1, $\cdots$, $(m-i,2^{m-i}-(m-i))$ is decomposed into $(1,2^{m-(i+1)}-1)$ and $(m-(i+1),2^{m-(i+1)}-(m-(i+1)))$ on the recursion level i and so on. Further, $(m-(m-2),2^{m-(m-2)}-(m-(m-2)))=(2,2)$ is decomposed into $(1,2^{m-(m-1)}-1)=(1,1)$ and $(m-(m-1),2^{m-(m-1)}-(m-(m-1)))=(1,1)$ on the recursion level $m-2$ where $(1,2^{m-(m-1)}-1)=(1,1)$ and $(m-(m-1),2^{m-(m-1)}-(m-(m-1)))=(1,1)$ correspond the last two input pairs on the recursion level $m-1$. □

From theorem 8 we have $n_1^1=2^{m-1}-1$, $n_1^2=2^{m-2}-1$, $\cdots$, $n_1^{m-2}=2^{m-(m-2)}-1$, $n_1^{m-1}=2^{m-(m-1)}-1$, $n_2^{m-1}=2^{m-(m-1)}-1$. Therefore we can see that it holds $n_1^1+n_1^2+\cdots+n_1^{m-1}+n_2^{m-1}=2^{m-1}-1+2^{m-2}-1+\cdots+2^2-1+2^1-1+1=2(2^{m-1}-1)-(m-1)+1=2^m-m=n$.

Further the number of the required comparisons for merging of maximum spanning case can be calculated exactly as follows:

Theorem 9. Let any input pair (m,n) with $m=2^k$ and $n=2^m-m$ of maximum spanning case be given. Then SymMerge requires exactly $\frac{1}{2}m^2+(m+1)\log m-\frac{3}{2}m+2$ comparisons for merging input pair (m,n).

Proof. Since $m=2^k$ and $n=2^m-m$, the overall number of the required comparisons for merging (m,n) is $\sum_{i=1}^{m}(\lfloor\log i\rfloor+1)+\sum_{i=1}^{m-1}(\lfloor\log(2^i-1)\rfloor+1)$. Since $\sum_{i=1}^{m}(\lfloor\log i\rfloor+1)=\sum_{i=1}^{2^k}(\lfloor\log i\rfloor+1)=k+2^k(k-1)+2$ and $\sum_{i=1}^{m-1}(\lfloor\log(2^i-1)\rfloor+1)=(m-2)+(m-3)+\cdots+1+(m-1)=\frac{m(m-1)}{2}$, we have $k+2^k(k-1)+2+\frac{m(m-1)}{2}=\log m+m(\log m-1)+2+\frac{1}{2}m^2-\frac{1}{2}m=\frac{1}{2}m^2+(m+1)\log m-\frac{3}{2}m+2$. □

4. Complexity Analysis of SymMerge Algorithm using Recurrence Relation

In this section we show the complexity for the special cases "symmetric case and maximum spanning case" regarding to the number of comparisons can be defined by recurrence relations.

Let $T_s(m,n)$ denote the overall required number of comparisons for any input pair (m,n) of the symmetric case. Then $T_s(m,n)$ is given as follows:

Definition 10. The complexity $T_s(m,n), m \leq n$ is defined recursively as follows:

$$T_s(1,n) = \lfloor\log n\rfloor+1$$
$$T_s(m,n) = \lfloor\log m\rfloor+1+T_s(\left\lfloor\frac{m}{2}\right\rfloor,\left\lfloor\frac{n}{2}\right\rfloor)+T_s(\left\lceil\frac{m}{2}\right\rceil,\left\lceil\frac{n}{2}\right\rceil)$$

Let $T_m(m,n)$ denote the overall required number of comparisons for any input pair (m,n) of the maximum spanning case. Then $T_m(m,n)$ is given as follows:

Definition 11. The complexity $T_m(m,n), m \leq n$ is defined recursively as follows:

$$T_m(1,n) = \lfloor \log n \rfloor + 1$$
$$T_m(m,n) = \lfloor \log m \rfloor + 1 + T_m(1, \left\lfloor \frac{n-m}{2} \right\rfloor + m - 1) + T_m(m-1, \left\lceil \frac{n-m}{2} \right\rceil + 1)$$

Now, let $T_{0,m}(m,n)$ denote the overall required number of comparisons for input (m,n) of the case such that on the recursion level 0 (m,n) is partitioned to $(0(=m_1^1), n_1^1)$ and $(m(=m_2^1), n_2^1)$ where $(m(=m_2^1), n_2^1)$ corresponds with the maximum special case. Then it holds $T_{0,m}(m,n) = \lfloor \log m \rfloor + 1 + T_m(m, \left\lceil \frac{n-m}{2} \right\rceil)$. In the following we show that it holds $T_{0,m}(m,n) \le T_m(m,n)$ for any input (m,n).

Theorem 8. For any $(m,n), m \le n, T_{0,m}(m,n) \le T_m(m,n)$.

Proof. If $m = 1$, then $T_{0,m}(1,n) = \lfloor \log n \rfloor + 1 \le T_m(1,n)$. If $m \ge 2$, then the following holds:

$$\begin{aligned} T_{0,m}(m,n) &= \lfloor \log m \rfloor + 1 + T_m(m, \left\lceil \frac{n-m}{2} \right\rceil) \\ &= \lfloor \log m \rfloor + 1 + T_m(1, \left\lfloor \frac{\beta - m}{2} \right\rfloor + m - 1) \\ &\quad + T_m(m-1, \left\lceil \frac{\beta - m}{2} \right\rceil + 1) \\ &\quad Here, let\ \beta = \left\lceil \frac{n-m}{2} \right\rceil \\ T_m(m,n) &= \lfloor \log m \rfloor + 1 + T_m(1, \left\lfloor \frac{n-m}{2} \right\rfloor + m - 1) \\ &\quad + T_m(m-1, \left\lceil \frac{n-m}{2} \right\rceil + 1) \end{aligned}$$

Since

$$T_m(1, \left\lceil \frac{\beta - m}{2} \right\rceil + m - 1) = \left\lfloor \log(\left\lceil \frac{\beta - m}{2} \right\rceil + m - 1) \right\rfloor + 1$$
$$\le \left\lfloor \log(\left\lfloor \frac{n-m}{2} \right\rfloor + m - 1) \right\rfloor + 1 = T_m(1, \left\lfloor \frac{n-m}{2} \right\rfloor + m - 1)$$

and

$$T_m(m-1, \left\lceil \frac{\beta - m}{2} \right\rceil + 1 \le T_m(m-1, \left\lceil \frac{n-m}{2} \right\rceil + 1,$$

we have $T_{0,m}(m,n) \le T_m(m,n)$.

Further, by applying theorem 8, the definition $T_m(m,n)$ with $n = 2^m - m$ can be simplified as follows:
Let $T_{msc}(m, 2^m - m)$ denote the overall required number of comparisons for any input pair $(m, 2^m - m)$ of the maximum spanning case, i.e. $T_{msc}(m, 2^m - m)$ is a simplified definition of $T_m(m, 2^m - m)$.

Definition 12. The complexity $T_{msc}(m, 2^m - m)$ is defined recursively as follows:

$$T_{msc}(1,1) = 1$$
$$T_{msc}(m, 2^m - m) = \lfloor \log m \rfloor + 1 + \lfloor \log(2^{m-1} - 1) \rfloor + 1 + T_{msc}(m-1, 2^{m-1} - (m-1))$$

5. Conclusion

We showed that the SymMerge algorithm requires exactly $m \log \frac{n}{m} + 4m - \log m - 3$ comparisons for symmetric case of $m = 2^k, n = 2^l, l \ge k$ and exactly $\frac{1}{2}m^2 + (m+1)\log m - \frac{3}{2}m + 2$ comparisons for maximum spanning case of $m = 2^k, n = 2^m - m$ where m and n represent the sizes of input sequences u and v. Additionally we showed that the complexity of the Symmerge algorithm regarding to the overall required number of comparisons for these special cases can be defined by recurrence relations.

Based on this study, the least upper bound for complexity of the SymMerge algorithm may be calculated. We conjecture that if $m = 2^k, n = 2^m - m$, then the complexity of the SymMerge algorithm has the least upper bound $m \log \frac{n}{m} + 4m - \log m - 3$ in general.

References

[1] K. Dudzinski and A. Dydek. "On a stable storage merging algorithm." *Information Processing Letters*, Vol. 12, pp. 5-8, February 1981.

[2] V. Geert, J. Katajainen, T. Pasanen, "Asymptotically ecient in-place merging." *Theoretical Computer Science* 237 (1/2), pp. 159-181, 2000.

[3] J. Chen, "Optimizing stable in-place merging." *Theoretical Computer Science* 302 (1/3), pp. 191-210, 2003.

[4] P. S. Kim and A. Kutzner, "Ratio Based Stable in-Place Merging," *TAMC 2008, Lecture Notes in Computer Science, Springer*, Vol. 4978, pp. 246-257, 2008.

[5] P. S. Kim and A. Kutzner, "Stable Minimum Storage Merging by Symmetric Comparisons," *ESA 2004, Lecture Notes in Computer Science, Springer*, Vol. 3221, pp. 714-723, 2004.

[6] P. S. Kim and A. Kutzner, "On a Simple and Stable Merging Algorithm," *Journal of The Korean Institute of Intelligent Systems*, Vol. 20, No. 4, pp. 455-462,

2010.
[7] P. S. Kim, "Complexity of the Symmerge Algorithm," *Journal of The Korean Institute of Intelligent Systems*, Vol. 18, No. 2, pp. 272-277, 2008.

저 자 소 개

김복선(Pok-Son Kim)
2002년~현재 : Professor,
Department of Mathematics,
Kookmin University

관심분야 : Merging Algorithms, Scheduling Problems, Complexity Analysis
Phone : +82-2-910-4747
E-mail : pskim@kookmin.ac.kr